A One-Dimensional Bin Packing Problem with Shelf Divisions *

E. C. Xavier a,* F. K. Miyazawa a

aInstituto de Computação — Universidade Estadual de Campinas, Avenida Albert Einstein, 1251, Caixa Postal 6176 — 13084-971 — Campinas-SP — Brazil, Fax: (+55)(19)3521-5847

Abstract

Given bins of size B, non-negative values d and Δ, and a list L of items, each item e ∈ L with size se and class ce, we define a shelf as a subset of items packed inside a bin with total items size at most Δ such that all items in this shelf have the same class. Two subsequent shelves must be separated by a shelf division of size d. The size of a shelf is the total size of its items plus the size of the shelf division. The Class Constrained Shelf Bin Packing Problem (CCSBP) is to pack the items of L into the minimum number of bins, such that the items are divided into shelves and the total size of the shelves in a bin is at most B. We present hybrid algorithms based on the First Fit (Decreasing) and Best Fit (Decreasing) algorithms, and an APTAS for the problem CCSBP when the number of different classes is bounded by a constant C.

Key words: Approximation algorithms, bin packing, shelf packing

1991 MSC: 68W25

* An extended abstract of this paper was presented at GRACO2005 (2nd Brazilian Symposium on Graphs, Algorithms, and Combinatorics) and appeared in Electronic Notes in Discrete Mathematics 19 (2005) 329–335.

This research was partially supported by CNPq (471460/04-4, 478818/03-3, 478470/06-1, 306526/04-2 and 490333/04-4) and ProNEx—FAPESP/CNPq (Proc. 2003/09925-5).

* Corresponding author: Instituto de Computação — Universidade Estadual de Campinas, Caixa Postal 6176 — 13084-971 — Campinas-SP — Brazil, Fax: (+55)(19)3521-5847

Email addresses: eduardo.xavier@gmail.com (E. C. Xavier ), fkm@ic.unicamp.br (F. K. Miyazawa).

Preprint submitted to Discrete Applied Mathematics 12 December 2006
1 Introduction

In this paper we present approximation algorithms for a class constrained bin packing problem when the items must be separated by non-null shelf divisions. We denote this problem by Class Constrained Shelf Bin Packing Problem (CCSBP).

An instance for the CCSBP problem is a tuple $I = (L, s, c, d, \Delta, B)$, where $L$ is a list of items, $s$ and $c$ are size and class functions over $L$, $d$ is the size of the shelf division, $\Delta$ is the maximum size of a shelf and $B$ is the size of the bins. Given a sublist of items $L' \subseteq L$ we denote by $s(L)$ the sum of the sizes of the items in $L'$, i.e., $s(L') = \sum_{e \in L'} s_e$. A shelf packing $\mathcal{P}$ of an instance $I$ for the CCSBP problem is a set of bins $\mathcal{P} = \{P_1, \ldots, P_k\}$, where the items packed in a bin $P_i \in \mathcal{P}$ are partitioned into shelves $\{N^i_1, \ldots, N^i q_i\}$ such that for each shelf $N^i_j$ we have that $s(N^i_j) \leq \Delta$, all items in $N^i_j$ are of the same class and $\sum_{j=1}^{q_i}(s(N^i_j) + d) \leq B$. Without loss of generality we consider that $0 \leq s_e \leq \Delta$ and $c_e \in \mathbb{Z}^+$ for each $e \in L$.

The CCSBP problem is to find a shelf packing of the items of $L$ into the minimum number of bins. This problem is NP-hard since it is a generalization of the bin packing problem: in this case consider that the instance has just one class, $\Delta = B$ and $d = 0$. We note that the term shelf is used under another context in the literature for the 2-D strip packing problem. In this case, packings are two staged packings divided into levels.

There are many practical applications for the CCSBP problem even when there is only one class of items. For example, when the items to be packed must be separated by non-null shelf divisions (inside a bin) and each shelf has a limited capacity. In Figure 1 we can see an example of a shelf packing of items into one bin, with $B = 60$, $\Delta = 17$, $d = 3$ and all items of the same class. The CCSBP problem is also adequate when some items cannot be stored in a same shelf (like foods and chemical products). In most of the cases, the sizes of the shelf divisions have non-negligible width. Although these problems are very common in practice, to our knowledge this is the first paper that presents approximation results for them.

An interesting application for the CCSBP problem was introduced by Ferreira et al. [4] in the iron and steel industry. In this problem, we have raw material rolls that must be cut into final rolls grouped by certain properties after two cutting phases. The rolls obtained after the first phase, called primary rolls, are submitted to different processing operations (tensioning, tempering, laminating, hardening etc.) before the second phase cut. Due to technological limitations, primary rolls have a maximum allowable width and each cut has a trimming process that generates a loss in the roll width. Each processing
operation has a high cost which implies items to be grouped before doing it, where each group corresponds to one shelf.

Given an algorithm $\mathcal{A}$, and an instance $I$ for the CCSBP problem, we denote by $\mathcal{A}(I)$ the number of bins used by algorithm $\mathcal{A}$ to pack the instance $I$ and by $\text{OPT}(I)$ the number of bins used in an optimal solution. The algorithm $\mathcal{A}$ is an $\alpha$-approximation, if $\mathcal{A}(I)/\text{OPT}(I) \leq \alpha$, for any instance $I$. In this case, we also say that $\mathcal{A}$ has an absolute performance bound $\alpha$. In bin packing problems, it is also usual to use the asymptotic worst case analysis. We say that $\mathcal{A}$ has an asymptotic performance bound $\alpha$ if there is a constant $\beta$ such that $\mathcal{A}(I) \leq \alpha \text{OPT}(I) + \beta$ for any instance $I$.

Given an algorithm $A_\varepsilon$, for some $\varepsilon > 0$, and an instance $I$ for some problem $P$ we denote by $A_\varepsilon(I)$ the value of the solution returned by algorithm $A_\varepsilon$ when executed on instance $I$. We say that $A_\varepsilon$, for $\varepsilon > 0$, is an asymptotic polynomial time approximation scheme (APTAS) for the problem CCSBP if there exist constants $t$ and $K$ such that $A_\varepsilon(I) \leq (1 + t\varepsilon)\text{OPT}(I) + K$ for any instance $I$.

**Results:** In this paper we present hybrid algorithms for the CCSBP problem, based on the First Fit (Decreasing) and Best Fit (Decreasing) algorithms for the bin packing problem. When the number of different classes is bounded by a constant, we show that the hybrid versions of the First Fit and Best Fit algorithms have an asymptotic performance bound of 3.4 and the hybrid versions of the First Fit Decreasing and Best Fit Decreasing algorithms have an asymptotic performance bound less than 2.445. In the case where the number of different classes is part of the input, we show that the hybrid versions of the First Fit and Best Fit algorithms have an absolute performance bound of 4 and the hybrid version of the First Fit Decreasing algorithm has an absolute performance bound of 3. At last, for the case when the number of classes is bounded by a constant, we present an APTAS for the CCSBP problem.

**Related Work:** A special case of the CCSBP problem is the Bin Packing problem, which is one of the most studied problems in the literature. Some of the most famous algorithms for the bin packing problem are the algorithms FF, BF, FFD and BFD, with asymptotic performance bounds $17/10$, $17/10$, $
11/9 and 11/9, respectively. Fernández de la Vega and Lueker [3] presented an APTAS for the bin packing problem. Dawande et al. [2], presented approximation schemes for a class constrained version of the bin packing (CCBP), where bins can have different sizes and each bin is used to pack items of at most $k$ different classes, and the number of different classes in the input instance is bounded by a constant. Shachnai and Tamir [7], presented a polynomial time approximation scheme for a dual version of the problem CCBP also for the case where the number of different classes in the input instance is bounded by a constant. Shachnai and Tamir [8], considered a special case of an online class constrained bin packing problem. In this case all items have the same size and must be packed without knowledge of the next subsequent items of the input. We refer the reader to Coffman et al. [1] for a survey on approximation algorithms for bin packing problems. In [11], we consider the knapsack version of the CCSBP problem, where each item $e$ has also a value $v_e$. The objective is to find a shelf packing of a subset $S$ of the items in just one knapsack (bin) of size $K$, such that the total value of the items in $S$ is maximum. We also give a PTAS for this problem. We remark that, despite of the similarity of the problems, the techniques and algorithms used in this paper are not related to the ones used in the knapsack version of the problem. Practical approaches for the CCSBP problem were considered by Ferreira et al. [4], that introduced the problem in the iron and steel industry. Recently, the problem was considered by Hoto et al. [5] and Marques and Arenales [6]. Hoto et al. [5], considered the cutting stock version of the problem where a demand of items must be attended by the minimum number of bins. They use a column generation strategy. In [6] exact and heuristic algorithms are presented for a knapsack version of the problem.

1.1 Notation

On the forthcoming sections we use the following notation: Let $I = (L, s, c, d, \Delta, B)$ be an instance for the CCSBP problem. We denote by $n = |L|$ the number of items in this instance. For any integer $t$, we denote by $[t]$ the set $\{1, \ldots, t\}$. We assume that each class belongs to the set $[C]$. We assume that $C$ is bounded by a constant, unless otherwise stated. We denote by $OPT_s(I)$ the minimum number of non-null shelves in an optimal packing of $I$, and by $OPT(I)$ the number of bins in this optimal solution. Given a packing $P = \{P_1, \ldots, P_k\}$, we denote by $|P| = k$ the number of bins used in this packing, and by $N_s(P)$ the number of shelves used in all bins of $P$. Given an algorithm $A$ we denote by $A(I)$ the number of bins used by the algorithm $A$ to pack the instance $I$. 

4
1.2 Simple Lower Bounds

The following facts present lower bounds for the number of bins used in any optimum solution for the CCSBP problem.

**Fact 1** For any instance \( I = (L, s, c, d, \Delta, B) \), we have

\[
\text{OPT}(I) \geq \frac{s(L)}{\lceil B/(d+\Delta) \rceil \Delta}.
\]

**Proof.** Since \( \lceil B/(d+\Delta) \rceil \) is an upper bound for the number of totally filled shelves in a bin, the total items size in a bin is at most \( B/(d+\Delta) \). \( \square \)

**Fact 2** For any instance \( I = (L, s, c, d, \Delta, B) \), we have

\[
\text{OPT}(I) \geq \frac{s(L) + \text{OPT}_s(I)d}{B} \geq \frac{s(L) + \lceil s(L)/\Delta \rceil d}{B}.
\]

**Proof.** The statement holds since \( \lceil s(L)/\Delta \rceil \) is a lower bound for the number of shelves used in any packing. \( \square \)

2 Hybrid versions of the First Fit and Best Fit Algorithms

In this section we present hybrid versions of the First Fit (Decreasing) and Best Fit (Decreasing) algorithms, for the classic bin packing problem, to the CCSBP problem. Without loss of generality, we assume that all bins have capacity 1.

We briefly describe how these algorithms work for the classic bin packing problem. The First Fit (FF) and the Best Fit (BF) algorithms pack the items of a given list \( L = (e_1, \ldots, e_m) \) in the order given by \( L \). Assume that the items \( e_1, \ldots, e_{i-1} \) have been packed into bins \( B_1, B_2, \ldots, B_k \), each bin with capacity 1. To pack the next item \( e_i \), the algorithm FF (resp. BF) finds the smallest index \( j, 1 \leq j \leq k \), such that \( s(B_j) + s(e_i) \leq 1 \) (resp. \( s(B_j) \) is maximum given that \( s(B_j) + s(e_i) \leq 1 \)). If the algorithm FF (resp. BF) finds such a bin, the item \( e_i \) is packed into the bin \( B_j \). Otherwise, the item \( e_i \) is packed into a new bin \( B_{k+1} \). This process is repeated until all the items of \( L \) have been packed.

The First Fit Decreasing (FFD) (resp. Best Fit Decreasing (BFD)) algorithm first sorts the items of \( L \) in non-increasing order of size and then apply the algorithm FF (resp. BF). The following result holds (see [1,9]).
**Theorem 3** For any instance $I$ for the bin packing problem, we have

$$
\text{FF}(I) \leq \frac{17}{10} \text{OPT}(I) + 1, \quad \text{BF}(I) \leq \frac{17}{10} \text{OPT}(I) + 1,
$$
$$
\text{FFD}(I) \leq \frac{11}{9} \text{OPT}(I) + 3, \quad \text{BFD}(I) \leq \frac{11}{9} \text{OPT}(I) + 3
$$

and

$$
\text{FFD}(I) \leq \frac{3}{2} \text{OPT}(I).
$$

Now we can present the hybrid algorithms for the problem CCSBP.

**Algorithms SFF, SBF, SFFD and SBF:** Given an instance $I = (L, s, c, d, \Delta, B)$, the algorithm SFF (resp. SBF, SFFD and SBF) uses the algorithm FF (resp. BF, FFD and BFD) to pack all items of a same class into shelves of size $\Delta$. The algorithm considers the size of each generated shelf as the total items size in the shelf plus the size of the shelf division $d$. The set of generated shelves are then packed into bins of size $B$ using the algorithm FF (resp. BF, FFD and BFD).

Given an instance $I = (L, s, c, d, \Delta, B)$ for the CCSBP problem, we denote by $\text{OPT}_\Delta(I)$ the minimum number of shelves of size $\Delta$ needed to pack $L$, where all items in a shelf have the same class. Clearly $\text{OPT}_\Delta(I)$ is a lower bound for the number of shelves used in any optimal solution. That is, $\text{OPT}_\Delta(I) \leq \text{OPT}_s(I)$.

**Theorem 2.1** Let $I$ be an instance for the CCSBP problem. If the number of classes in $I$ is bounded by $C$ then

$$
\text{SFF}(I) \leq (3 + \frac{2}{3}) \text{OPT}(I) + 2C, \quad \text{SBF}(I) \leq (3 + \frac{2}{3}) \text{OPT}(I) + 2C,
$$
$$
\text{SFFD}(I) \leq (2 + \frac{4}{9}) \text{OPT}(I) + 6C \quad \text{and} \quad \text{SBFD}(I) \leq (2 + \frac{4}{9}) \text{OPT}(I) + 6C.
$$

**PROOF.** Let $\mathcal{A}'$ be an algorithm in $\{\text{FF, BF, FFD, BFD}\}$ such that

$$
\mathcal{A}'(I') \leq \alpha \text{OPT}(I') + \beta
$$

for any instance $I'$ of the classic bin packing problem and let $\mathcal{A}$ be the corresponding algorithm in $\{\text{SFF, SBF, SFFD, SBFD}\}$. Let $I = (L, s, c, d, \Delta, B)$ be an instance for the CCSBP problem. Consider the packing $\mathcal{P}$ produced by the algorithm $\mathcal{A}$ for the instance $I$. We consider that $|\mathcal{P}| > 1$, otherwise $\mathcal{P}$ is optimum. On average, all bins in $\mathcal{P}$ are filled by at least $1/2$ (including shelf divisions), since the algorithms pack the shelves in such a way that any pair of bins have total contents size greater than $1$. We can conclude the following:
\( A(I)(1/2) \leq s(L) + N_s(P)d \)
\( \leq s(L) + (\alpha \text{OPT}_\Delta(I) + C\beta)d \)
\( \leq s(L) + (\alpha \text{OPT}_s(I) + C\beta)d \) (1)
\( \leq \alpha(s(L) + d\text{OPT}_s(I)) + C\beta d \)
\( \leq \alpha \text{OPT}(I) + C\beta, \) (2)

\( \) where (1) holds from Theorem 3, and (3) follows from Fact 2 and the fact that \( d \leq 1. \) \( \Box \)

Notice that any algorithm for the classic bin packing problem can be easily extended to an algorithm with the same asymptotic performance bound and that produces bins that on average are filled by at least half of its capacities. Therefore, the following result can be easily derived as a generalization of the previous theorem.

**Corollary 2.2** Given an algorithm \( A' \) for the bin packing problem, where \( A'(L) \leq \alpha \text{OPT}(L) + \beta \) for any instance \( L, \) then there exists an algorithm \( A \) for the CCSBP such that
\[
A(I) \leq 2\alpha \text{OPT}(I) + 2\beta C,
\]

for any instance \( I \) of the CCSBP problem.

This result shows that when the number of classes is bounded by a constant we can obtain, using an APTAS for the bin packing problem, algorithms for the CCSBP problem with asymptotic performance bound as close to 2 as desired (although with high time complexity and with high value of \( \beta \)).

Now we consider that the number of different classes is not bounded by a constant. Notice that if a given algorithm \( A' \) for the bin packing problem has absolute performance bound \( \alpha, \) then we can derive an algorithm \( A \) for the CCSBP problem with absolute performance bound \( 2\alpha, \) even if the number of different classes of items is given as part of the input. Using the fact that algorithms FF, BF and FFD have absolute performance bound 2, 2 and 3/2 respectively, we can obtain the following result.

**Corollary 2.3** Let \( I \) be an instance for the CCSBP problem, then
\[
\text{SFF}(I) \leq 4\text{OPT}(I), \quad \text{SBF}(I) \leq 4\text{OPT}(I) \quad \text{and} \quad \text{SFFD}(I) \leq 3\text{OPT}(I),
\]

even if the number of different classes is not bounded by a constant.

From the practical point of view, the size of the shelf division \( d \) is not so large compared with \( \Delta. \) The next theorem shows that if \( d \) is a small fraction
of $\Delta$, we can obtain a better performance bound for the Best and First Fit strategies.

**Theorem 2.4** Let $I = (L, s, c, d, \Delta, B)$ be an instance for the CCSBP problem. If the number of classes in $I$ is bounded by $C$ and $d = \frac{\Delta}{r}$, $r \geq 1$, we have

$$SFF(I) \leq \left(2 + \frac{14}{5r}\right) \text{OPT}(I) + 2C,$$
$$SBF(I) \leq \left(2 + \frac{14}{5r}\right) \text{OPT}(I) + 2C,$$
$$SFFD(I) \leq \left(2 + \frac{8}{9r}\right) \text{OPT}(I) + 6C,$$
$$SBFD(I) \leq \left(2 + \frac{8}{9r}\right) \text{OPT}(I) + 6C,$$

and

$$SFFD(I) \leq \left(2 + \frac{2}{r}\right) \text{OPT}(I).$$

**PROOF.** Let $\mathcal{A}'$ be an algorithm in $\{FF, BF, FFD, BFD\}$ such that

$$\mathcal{A}'(I') \leq \alpha \text{OPT}(I') + \beta$$

for any instance $I'$ of the classic bin packing problem and let $\mathcal{A}$ the corresponding algorithm in $\{SFF, SBF, SFFD, SBFD\}$. Let $I = (L, s, c, d, \Delta, B)$ be an instance for the CCSBP problem and $\mathcal{P}$ the packing produced by the algorithm $\mathcal{A}$ for the instance $I$.

We divide the proof in two cases, according to the values of $N_s(\mathcal{P})$ and $\text{OPT}_s(I)$.

**Case 1:** $N_s(\mathcal{P}) < \text{OPT}_s(I)$. In this case, we have

$$\mathcal{A}(I)(1/2) \leq s(L) + N_s(\mathcal{P})d$$
$$< s(L) + \text{OPT}_s(I)d$$
$$\leq \text{OPT}(I),$$

where inequality (4) holds from Fact 2. That is,

$$\mathcal{A}(I) \leq 2\text{OPT}(I).$$

**Case 2:** $N_s(\mathcal{P}) \geq \text{OPT}_s(I)$. In this case, we can follow the proof of Theorem 2.1 and obtain inequality (2). That is,

$$\mathcal{A}(I)(1/2) \leq s(L) + (\alpha \text{OPT}_s(I) + C/\beta)d.$$

Since on average each shelf generated by the algorithm $\mathcal{A}$ is filled by at least $\Delta/2$ (not including the shelf division), we have
\[
\text{OPT}(I) \geq s(L) \\
\geq N_s(\mathcal{P})(\Delta/2) \\
\geq \text{OPT}_s(I) \Delta/2 = \text{OPT}_s(I)dr/2.
\]

That is, \( \text{OPT}_s(I)d \leq (2\text{OPT}(I))/r \). Therefore, from inequality (6), we have

\[
\mathcal{A}(I)(1/2) \leq s(L) + (\alpha \text{OPT}_s(I) + C\beta)d. \\
= s(L) + \text{OPT}_s(I)d + (\alpha - 1)\text{OPT}_s(I)d + C\beta d. \\
\leq \text{OPT}(I) + \frac{\alpha - 1}{r}(2\text{OPT}(I)) + C\beta d. \\
\leq (1 + \frac{2(\alpha - 1)}{r})\text{OPT}(I) + C\beta d.
\]

That is,

\[
\mathcal{A}(I) \leq (2 + \frac{4(\alpha - 1)}{r})\text{OPT}(I) + 2\beta C. \tag{7}
\]

The theorem follows from inequalities (5), (7) and theorem 3. \( \square \)

The following proposition shows that the previous theorem presents an asymptotic performance bound that is tight for the algorithms SFF and SBF, when \( d \) is very small compared to \( \Delta \).

**Proposition 2.5** The asymptotic performance bound of the algorithms SFF and SBF is at least 2, even when there is only one class.

**Proof.** Let \( I_n = (L, s, c, d, \Delta, B) \) be an instance with \( L = (e_1, \ldots, e_{2n}) \), \( \varepsilon = 1/n \), \( d = \varepsilon/2 \), \( B = 1 \), \( \Delta = 1/2 \) and \( s(e_i) = 1/2 - \varepsilon \) when \( i \) is odd and \( s(e_i) = \varepsilon \) otherwise. Notice that \( d = \Delta/n \). Also assume that all items have a same class. The SFF and SBF algorithms applied over this instance generates \( n \) shelves, each one containing one item of size \( 1/2 - \varepsilon \) and one item of size \( \varepsilon \). The final packing generated by these algorithms has \( n \) bins, each one containing one shelf. An optimal packing with \( n/2 + 1 \) bins can be obtained in such a way that \( n/2 \) bins have two shelves each, one shelf with an item of size \( 1/2 - \varepsilon \), and the other shelf with two items, one item of size \( 1/2 - \varepsilon \) and another of size \( \varepsilon \). The last bin contains the remaining items of size \( \varepsilon \). \( \square \)
3 An Asymptotic Polynomial Time Approximation Scheme

In this section we present an APTAS for the CCSBP problem when the number of different classes is bounded by a constant $C$.

The algorithm is presented in Figure 2 and is denoted by $\text{ASBP}_\varepsilon$. It considers two cases: When $\varepsilon \geq d + \Delta$, it uses an algorithm denoted by $\text{ASBP}_\varepsilon'$ and in the other case, it uses an algorithm denoted by $\text{ASBP}_\varepsilon''$. Notice that the algorithm $\text{ASBP}_\varepsilon''$ receives as input a rescaled instance so that the maximum shelf capacity is 1.

\begin{algorithm}
\textbf{Input:} List of items $L$, each item $e \in L$ with size $s_e$ and class $c_e$, maximum capacity of a shelf $\Delta$, shelf divisions of size $d$, bins of capacity $B = 1$.
\textbf{Output:} Shelf packing $\mathcal{P}$ of $L$.
\textbf{Subroutines:} Algorithms $\text{ASBP}_\varepsilon'$ and $\text{ASBP}_\varepsilon''$.
\begin{enumerate}
  \item If $\varepsilon \geq d + \Delta$ then
  \item \quad $\mathcal{P} \leftarrow \text{ASBP}_\varepsilon'(L, s, c, d, \Delta, B)$
  \item \quad else
  \item \quad Scale the sizes $d$, $\Delta$, $B$ and $s_e$, for each $e \in L$, proportionally so that $\Delta = 1$.
  \item \quad // The condition to enter in this case is now equivalent to $\varepsilon \leq (d+\Delta)/B$.
  \item \quad $\mathcal{P} \leftarrow \text{ASBP}_\varepsilon''(L, s, c, d, \Delta, B)$.
  \item \quad Return $\mathcal{P}$.
\end{enumerate}
\end{algorithm}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{algorithm.pdf}
\caption{Algorithm $\text{ASBP}_\varepsilon$.}
\end{figure}

The intuition to consider these two cases is that in the first case, we can pack shelves almost optimally because the maximum size of a shelf is bounded by $\varepsilon$, and then the bins can be filled by at least $(1 - \varepsilon)$. In the second case, since $\varepsilon < d + \Delta$, we can bound by a constant the number of shelves used in each bin of an optimal solution. Then an enumeration step can be done to guess the shelves that are used in an optimal solution and an almost optimal shelf packing can be generated for large items. Small items are packed later using a linear programming strategy. In the following two subsections we show that algorithms $\text{ASBP}_\varepsilon'$ and $\text{ASBP}_\varepsilon''$ are APTAS.

3.1 The algorithm $\text{ASBP}_\varepsilon'$

In this section we show that the algorithm $\text{ASBP}_\varepsilon'$ is an APTAS for its corresponding case. This algorithm uses two subroutines: One is the FF algorithm and the other is an APTAS for the one dimensional bin packing problem presented by Fernandez de la Vega and Lueker [3]. We consider the version of this APTAS presented by Vazirani [10], which we denote by FL$_\varepsilon$, for which
the following statement holds.

**Theorem 3.1** For any \( \varepsilon > 0 \), there exists a polynomial time algorithm \( \text{FL}_\varepsilon \) to pack a list of items \( L \), each item \( e \in L \) with size \( s_e \in [0, \Delta] \), into bins of capacity \( \Delta \) such that \( \text{FL}_\varepsilon(L) \leq (1 + \varepsilon) \text{OPT}_\Delta(L) + 1 \), where \( \text{OPT}_\Delta(L) \) is the minimum number of bins of capacity \( \Delta \) to pack \( L \).

The algorithm \( \text{ASBP}'_\varepsilon \) is presented in Figure 3. Given an instance \( I \), the algorithm \( \text{ASBP}'_\varepsilon \) first packs all items of the instance into bins of size \( \Delta \) using the algorithm \( \text{FL}_\varepsilon \). The algorithm \( \text{ASBP}'_\varepsilon \) considers each one of these bins of size \( \Delta \) as a shelf, where the size of a shelf is its total items size plus the size \( d \) of a shelf division. The algorithm \( \text{ASBP}'_\varepsilon \) packs these shelves into bins of size 1 using the algorithm FF.

```
ALGORITHM ASBP'_\varepsilon(L, s, c, \Delta, d, B)
Input: List of items \( L \), each item \( e \in L \) with size \( s_e \) and class \( c_e \), maximum capacity of a shelf \( \Delta \), shelf divisions of size \( d \), bins of capacity \( B = 1 \) and \( \varepsilon \geq d + \Delta \).
Output: Shelf packing \( \mathcal{P} \) of \( L \).
Subroutines: Algorithms \( \text{FL}_\varepsilon \) and FF.
1. Let \( L_c \) be the set of items of class \( c \) in \( L \).
2. For each class \( c \in [C] \) let \( \mathcal{P}^\Delta_{c} \) be the packing of \( L_c \) obtained by the algorithm \( \text{FL}_\varepsilon \) using bins of capacity \( \Delta \).
3. Let \( \mathcal{P}^\Delta \) be the union of the packings \( \mathcal{P}^\Delta_{c} \), for each \( c \in [C] \).
4. Consider each bin \( D \in \mathcal{P}^\Delta \) as a shelf with size \( \sum_{e \in D} s_e + d \).
5. Let \( S \) be the set of shelves obtained from \( \mathcal{P}^\Delta \).
6. Let \( \mathcal{P} \) be the packing obtained with the algorithm FF to pack the shelves of \( S \) into unit bins.
7. Return \( \mathcal{P} \).
```

![Fig. 3. Algorithm ASBP'_\varepsilon where \( \varepsilon \geq d + \Delta \).](image)

The following statement holds for the algorithm \( \text{ASBP}'_\varepsilon \).

**Lemma 3.2** The algorithm \( \text{ASBP}'_\varepsilon \), is an APTAS for the CCSBP problem when the given instance \( I \) is such that \( B = 1 \) and \( \varepsilon \geq d + \Delta \).

**PROOF.** In step 2, the algorithm obtains a packing \( \mathcal{P}^\Delta_{c} \) of items of class \( c \) in \( L \) (items in \( L_c \)) into bins of capacity \( \Delta \) using the algorithm \( \text{FL}_\varepsilon \). By Theorem 3.1, we have

\[
|\mathcal{P}^\Delta_{c}| \leq (1 + \varepsilon) \text{OPT}_\Delta(L_c) + 1. 
\]  

(8)

The algorithm then considers each bin in \( \mathcal{P}^\Delta \) as a shelf and obtains a shelf packing \( \mathcal{P} \) using the algorithm FF to pack these shelves into unit bins. Since
\[ \varepsilon \geq d + \Delta, \text{ all bins of } \mathcal{P}, \text{ except perhaps the last, must be filled by at least } 1 - \varepsilon. \text{ So,} \]

\[
(\text{ASBP}'_\varepsilon(L) - 1)(1 - \varepsilon) \leq s(L) + d|\mathcal{P}_\Delta| \\
\leq s(L) + d \sum_{c=1}^{\ell} ((1 + \varepsilon)\text{OPT}_\Delta(L_c) + 1) \\
\leq (1 + \varepsilon)(s(L) + d \sum_{c=1}^{\ell} \text{OPT}_\Delta(L_c)) + dC \\
\leq (1 + \varepsilon)(s(L) + d\text{OPT}_s(I)) + dC \\
\leq (1 + \varepsilon)\text{OPT}(I) + C
\]

where inequality (9) is valid from Fact 2. Also notice that \( d < 1. \) Therefore, for any \( 0 < \varepsilon < 1/3 \) we have

\[
\text{ASBP}'_\varepsilon(L) \leq \frac{1 + \varepsilon}{1 - \varepsilon} \text{OPT}(I) + \frac{C}{1 - \varepsilon} + 1 \\
\leq (1 + 3\varepsilon)\text{OPT}(I) + \frac{3C}{2} + 1.
\]

Notice that the running time of the algorithm ASBP'_\varepsilon only depends on the running times of algorithms FL_\varepsilon and FF, and the value of \( C. \) Let \( T_{FL}(n, \varepsilon) \) and \( T_{FF}(n, \varepsilon) \) be the running times of algorithms FL_\varepsilon and FF respectively. The running time of algorithm ASBP'_\varepsilon is \( O(CT_{FL}(n, \varepsilon) + T_{FF}(n, \varepsilon)) \). Since the algorithms FL_\varepsilon and FF have polynomial time complexity in \( n \) for fixed \( \varepsilon, \) the complexity time of algorithm ASBP'_\varepsilon is also polynomial in \( n \) for fixed \( \varepsilon. \)

\[ \square \]

### 3.2 The algorithm ASBP''_\varepsilon

Now, assume that the algorithm ASBP_\varepsilon obtains a shelf packing with the algorithm ASBP''_\varepsilon. Throughout this section, we consider that \( s_\varepsilon, d, \Delta \) and \( B \) is the rescaled instance, such that \( \Delta = 1. \) Notice that, the equivalent condition to enter in this case is

\[ \varepsilon < \frac{d + \Delta}{B} = \frac{d + 1}{B}. \tag{10} \]

Notice that the maximum number of shelves completely filled packed in a bin is at most \( \left\lfloor \frac{B}{d + \Delta} \right\rfloor \) which from (10) is at most \( \frac{1}{\varepsilon} + 1. \) Observe that if there is any bin with more than \( \frac{2}{\varepsilon} + 2 \) shelves of a same class, it has at least two shelves of this class with total size at most \( \Delta. \) In this case, these two shelves can be
combined into only one shelf. Without loss of generality we consider that each
bin, in a solution for the CCSBP problem, contains at most $\frac{2}{\varepsilon} + 2$ shelves of
a same class.

In Figure 4 we present the algorithm ASBP$_\varepsilon''$. The algorithm first obtains a
pair $(\mathcal{P}_1, \mathcal{P})$ where $\mathcal{P}_1 \cup \mathcal{P}'$, for each $\mathcal{P}' \in \mathcal{P}$, is a packing of big items (items
with size at least $\varepsilon^2$). This pair is obtained by the subroutine $A_{LR}$. For each
packing $\mathcal{P}_1 \cup \mathcal{P}'$, $\mathcal{P}' \in \mathcal{P}$, the algorithm ASBP$_\varepsilon''$ uses the subroutine SMALL
to pack the items with size less than $\varepsilon^2$ into the packing $\mathcal{P}'$. At least one of
the generated packings uses at most $(1 + O(\varepsilon)) \text{OPT}(I) + O(1)$ bins as will be
shown latter. The algorithm returns the packing with the smallest number of
bins.

\begin{algorithm}
\textbf{Algorithm ASBP$_\varepsilon''$}($L, s, c, d, \Delta, B$)
\textbf{Input:} List of items $L$, each item $e \in L$ with size $s_e$ and class $c_e$, maximum
capacity of a shelf $\Delta = 1$, shelf divisions of size $d$, bins of capacity $B$
and $\varepsilon \leq (d + \Delta)/B$.
\textbf{Output:} Shelf packing $\mathcal{P}$ of $L$.
\textbf{Subroutines:} Algorithms $A_{LR}$ and SMALL.
1. Let $G$ be the set of items $e \in L$ with size $s_e \geq \varepsilon^2$ and $S$ the set $L \setminus G$.
2. Let $(\mathcal{P}_1, \mathcal{P})$ be a pair obtained from the algorithm $A_{LR}$ applied over the list
   $G$.
3. For each $Q \in \mathcal{P}$ do
   4. let $\hat{Q}$ be the packing obtained using the algorithm SMALL to pack $S$
      into $Q$.
5. Let $\mathcal{P}$ be a packing $\mathcal{P}_1 \cup \hat{Q}$ where $Q \in \mathcal{P}$ and $|\hat{Q}|$ is minimum.
6. Return $\mathcal{P}$.
\end{algorithm}

Fig. 4. Algorithm ASBP$_\varepsilon''$.

In the next subsections we present the subroutines used by the algorithm
ASBP$_\varepsilon''$. The first subroutine called $A_{LR}$ is used to generate a set of packings
of big items. On the next subsection we present an algorithm called SMALL
used to pack small items (items with size smaller than $\varepsilon^2$) in the packings
of the big items generated by the algorithm $A_{LR}$. In the last subsection we
present the analysis of the algorithm ASBP$_\varepsilon''$.

3.2.1 Generating Packings for the Big Items

In this section, we present the algorithm $A_{LR}$ used to pack items with size at
least $\varepsilon^2$ of a given input instance $I$. This algorithm generates a set of packings
such that at least one can be used to pack the small items, such that the
resulting packing has size at most $(1 + O(\varepsilon)) \text{OPT}(I) + O(1)$. This algorithm
uses the linear rounding technique, presented by Fernandez de la Vega and
Lueker [3], and considers only items with size at least $\varepsilon^2$. The algorithm $A_{LR}$
returns a pair \((\mathcal{P}_1, \mathcal{P})\), where \(\mathcal{P}_1\) is a packing for a list of very big items and \(\mathcal{P}\) is a set of packings for the remaining items.

We use the following notation in the description of the linear rounding technique: Given two lists of items \(X\) and \(Y\), let \(X_1, \ldots, X_C\) and \(Y_1, \ldots, Y_C\) be the partition of \(X\) and \(Y\) respectively in classes, where \(X_c\) and \(Y_c\) have only items of class \(c\) for each \(c \in [C]\). We write \(X \preceq Y\) if there is an injection \(f_c : X_c \to Y_c\) for each \(c \in [C]\) such that \(s(e) \leq s(f(e))\) for all \(e \in X_c\). Given two lists \(L_1\) and \(L_2\) we denote by \(L_1||L_2\) the concatenation of these lists.

The algorithm \(A_{LR}\) uses three subroutines: \(A_{ALL}\), \(SFF\), and \(A_R\). The algorithm \(SFF\) was presented in Section 2. In what follows we present the algorithms \(A_{ALL}\) and \(A_R\).

**Algorithm \(A_{ALL}\)**: This is an algorithm used as subroutine to generate all possible packings with at most \(\frac{2}{\varepsilon} + 2\) shelves of a same class, when the size of each item is bounded from below by a constant and the number of distinct sizes in each class is upper bounded by a constant \(t\). The algorithm may generate empty shelves (used latter to pack small items). The following lemma guarantees the existence of such an algorithm.

**Lemma 3.3** Given an instance \(I = (L, s, c, d, \Delta, B)\), with \(\Delta = 1\), where the number of distinct items sizes in each class is at most a constant \(t\), the number of different classes is bounded by a constant \(C\) and each item \(e \in L\) has size \(s_e \geq \varepsilon^2\), then there exists a polynomial time algorithm that generates all possible shelf packings of \(L\) with at most \(\frac{2}{\varepsilon} + 2\) shelves of a same class in each bin.

**Proof.** The number of items in a shelf is bounded by \(p = 1/\varepsilon^2\). Given a class, the number of different shelves for it is bounded by \(r = \left(\frac{p^{p+t}}{p}\right)\) and so, the number of different shelves is bounded by \(r = Cr^n\). Since the number of shelves in a bin is bounded by \(q = C(\frac{2}{\varepsilon} + 2)\), the number of different bins is bounded by \(u = \left(\frac{q^r}{q^n}\right)\). Notice that \(u\) is a (large) constant since all the values \(p, q, r\) and \(u\) depends only on \(\varepsilon, C\) and \(t\) which are constants.

Therefore, the number of all feasible packings is bounded by \(\left(\frac{n+u}{n}\right)^n\), which is bounded by \((n+u)^n\), which in turn is polynomial in \(n\). □

Notice that the complexity time of the algorithm \(A_{ALL}\) is \(O(n^{O(2C/\varepsilon)}\varepsilon^{O(1/\varepsilon^2)}\).

**Algorithm \(A_R\)**: Given two lists \(X\) and \(Y\) such that \(X \preceq Y\) and a packing \(\mathcal{P}_Y\) of \(Y\), there exists an algorithm, which we denoted by \(A_R\) (Replace), with
input \((\mathcal{P}_Y, X)\), that obtains a packing \(\mathcal{P}_X\) for \(X\) such that \(|\mathcal{P}_X| = |\mathcal{P}_Y|\) as the next lemma guarantees.

**Lemma 3.4** If \(X\) and \(Y\) are two lists with \(X \preceq Y\), then \(\text{OPT}(X) \leq \text{OPT}(Y)\). Moreover, if \(\mathcal{P}_Y\) is a shelf packing of \(Y\) then there exists a polynomial time algorithm \(A_R\) that given \(\mathcal{P}_Y\) obtains a shelf packing \(\mathcal{P}_X\) of \(X\) such that \(|\mathcal{P}_X| = |\mathcal{P}_Y|\).

**Proof.** The algorithm \(A_R\) sorts the lists \(X_c\) and \(Y_c\) for each \(c \in [C]\) in non-increasing order of items size and then replaces in this order, each item of \(Y_c\) in the packing \(\mathcal{P}_Y\) by an item of \(X_c\). The possible remaining items of \(Y_c\) are removed. \(\Box\)

For any instance \(X\), denote by \(\overline{X}\) the instance with precisely \(|X|\) items with size equal to the size of the smallest item in \(X\). Clearly, \(\overline{X} \preceq X\).

The algorithm \(A_{LR}\) is presented in Figure 5. It consists in the following: Let \(G_1, \ldots, G_C\) be the partition of the input list \(G\) into classes \(1, \ldots, C\) and let \(n_c = |G_c|\) for each class \(c\). The algorithm \(A_{LR}\) partition each list \(G_c\) into groups \(G^1_c, G^2_c, \ldots, G^k_c\). Let \(G^1 = \cup_{c=1}^C G^1_c\). The algorithm generates a packing \(\mathcal{P}_1\) of \(G^1\) using

\[O(\varepsilon)\text{OPT}(I) + 1\]

bins and a set \(\mathbb{P}\) with polynomial number of packings for the items in \(G \setminus G^1\). The packing \(\mathcal{P}_1\) is generated by the algorithm SFF and the set of packings \(\mathbb{P}\) is generated using the algorithms \(A_{ALL}\) and \(A_R\).

Denote by \(T_{ALL}, T_R\) and \(T_{SFF}\) the time complexity of algorithms \(A_{ALL}, A_R\), and SFF respectively. The time complexity of steps 1–3 of algorithm \(A_{LR}\) is bounded by \(O(n \log n)\). The overall time complexity of algorithm \(A_{LR}\) is \(O(n \log n + T_{SFF} + T_{ALL} + T_{ALL}T_R)\). Since \(T_{ALL}, T_R\) and \(T_{SFF}\) have polynomial time complexity in \(n\) for fixed \(\varepsilon\), the time complexity of algorithm \(A_{LR}\) is also polynomial in \(n\) for fixed \(\varepsilon\).

The following statement holds for the packing \(\mathcal{P}_1\).

**Lemma 3.5** The packing \(\mathcal{P}_1\) for the items in \(G^1\) is such that \(|\mathcal{P}_1| \leq 4\varepsilon \text{OPT}(I) + 1\).

**Proof.** First, consider the total items size packed in a bin \(T\) of some shelf packing. From Fact 1 any optimum solution must satisfy
Algorithm $A_{LR}(G)$

Input: List $G$ with $n$ items, each item $e \in G$ with size $s_e \geq \varepsilon^2$; maximum capacity of a shelf $\Delta = 1$; shelf divisions of size $d$ and bins of capacity $B$.

Output: A pair $(\mathcal{P}_1, \mathcal{P})$, where $\mathcal{P}_1$ is a packing and $\mathcal{P}$ is a set of packings, where $\mathcal{P}_1 \cup \mathcal{P}'$ is a packing of $G$ for each $\mathcal{P}' \in \mathcal{P}$.

Subroutines: Algorithms $A_{ALL}$, SFF and $A_R$.

1. Partition $G$ into lists $G_c$ for each class $c = 1, \ldots, C$ and let $n_c = |G_c|$, where $G^k_c = \{ G^j_c : j = 1, \ldots, k_c - 1 \}$ and $G^k_c = \{ G^j_c : j = 1, \ldots, k_c - 1 \}$.

2. Partition each list $G_c$ into $k_c \leq \lceil 1/\varepsilon^3 \rceil$ groups $G^1_c, G^2_c, \ldots, G^{k_c}_c$, such that $G^1_c \geq G^2_c \geq \cdots \geq G^{k_c}_c$.

3. Let $G^1 = \cup_{c=1}^{\mathcal{C}} G^1_c$.

4. Let $\mathcal{P}_1$ be a packing of $G^1$ obtained by the algorithm SFF.

5. Let $\mathcal{Q}$ be the set of all possible packings obtained with the algorithm $A_{ALL}$ over the list $(G^1_1 \ldots \lvert G^1_{k_c - 1} \rvert \ldots G^1_c \ldots \lvert G^k_{c - 1} \rvert \ldots G^k_c)$.

6. Let $\mathcal{P}$ be the set of packings obtained with the algorithm $A_R$ over each pair $(\mathcal{Q}, G^1_1 \ldots \lvert G^1_{k_c - 1} \rvert \ldots G^2_c \ldots \lvert G^k_{c - 1} \rvert \ldots G^k_c)$, where $\mathcal{Q} \in \mathcal{Q}$.

7. Return $(\mathcal{P}, \mathcal{P}_1)$.

Fig. 5. Algorithm to obtain packings for items with size at least $\varepsilon^2$.

\[
\text{OPT}(I) \geq \frac{s(L)}{|B/(d+\Delta)| \Delta} = \frac{s(L)}{|B/(d+1)|} \geq \frac{1}{2} \frac{s(L)}{B/(d+1)}. \tag{11}
\]

Notice that $\sum_{c=1}^{\mathcal{C}} n_c = n$. The algorithm SFF packs at least $\lfloor B/(d+\Delta) \rfloor$ shelves in each bin, each shelf with at least one item. This means that each bin has at least $\lfloor B/(d+\Delta) \rfloor$ items, except perhaps the last, each item with size at least $\varepsilon^2$ and at most 1. Since the group $G^1$ has at most $n\varepsilon^3$ items, the number of bins in the shelf packing $\mathcal{P}_1$ can be bounded as follows.

\[
|\mathcal{P}_1| \leq \left\lceil \frac{n\varepsilon^3}{|B/(d+\Delta)|} \right\rceil = \left\lceil \frac{n\varepsilon^3}{|B/(d+1)|} \right\rceil \\
\leq 2 \frac{n\varepsilon^3}{B/(d+1)} + 1 \leq 2 \varepsilon s(L) + 1 \leq 4\varepsilon \text{OPT}(I) + 1, \tag{12}
\]

where the inequality (12) is valid from (11). \qed
3.2.2 Packing the Small Items

In this section we present an algorithm to pack the small items. If we only consider the big items, at least one of the packings generated by the algorithm $A_{LR}$ has basically the same configuration of an optimal packing. That is, one of the generated packings has approximately the same number of bins and approximately the same shelves (including empty shelves that are used only for small items) of an optimal packing. Therefore the algorithm can guess how the small items are packed into the shelves of this packing, leaving only a small fraction of small items to be packed in new extra bins. Notice that a first approach to deal with the CCSBP problem, would be to produce the packing of the big items and then try to pack small items greedily. In the classic bin packing problem this approach works, since after packing the small items in the bins, each bin is filled by at least $(1 - \epsilon)$ of its capacity, except perhaps the last bin. In the CCSBP problem this strategy may not work, since after packing the small items, the packing could use more shelves. This way, each bin would not be filled with items by at least $(1 - \epsilon)$ of its capacity, since each bin also contains shelf divisions. To pack small items in the shelves generated by the algorithm $A_{LR}$ we use a linear programming strategy. This approach has an easier and clearer analysis leading to the APTAS.

The algorithm $\text{ASBP}_\varepsilon^{\prime}$ uses a subroutine denoted by SMALL to pack small items (size less than $\varepsilon^2$) into a given packing of big items. Let $P = \{P_1, \ldots, P_k\}$ be a shelf packing of a list of items $L$ and assume that we have to pack a set $S$ of small items, with size at most $\varepsilon^2$, into $P$. The packing of the small items is obtained from a solution of a linear program. Let $N_i^c$, $\ldots$, $N_{i+e}^c$ be the shelves of class $c$ in the bin $P_i$ of the packing $P$. For each shelf $N_j^c$, define a non-negative variable $x_{j}^{ic}$. The variable $x_{j}^{ic}$ indicates the total size of small items of class $c$ that is to be packed in the shelf $N_j^c$. Consider the following linear program denoted by LPS:

$$
\max \sum_{i=1}^{k} \sum_{c=1}^{C} \sum_{j=1}^{n_{ic}} x_{j}^{ic}
$$

subject to:

$$
s(N_j^c) + x_{j}^{ic} \leq \Delta \quad \forall \ i \in [k], \ c \in [C], \ j \in [n_{ic}], \ (1)
$$

$$
\sum_{c=1}^{C} \sum_{j=1}^{n_{ic}} s(N_j^c) + x_{j}^{ic} + d \leq B \quad \forall \ i \in [k], \ (2)
$$

$$
\sum_{i=1}^{t} \sum_{j=1}^{n_{ic}} x_{j}^{ic} \leq s(Sc) \quad \forall \ c \in [C], \ (3)
$$

$$
x_{j}^{ic} \geq 0 \quad \forall \ i \in [k], \ c \in [C], \ j \in [n_{ic}] \ (4)
$$

where $S_c$ is the set of small items of class $c$ in $S$.

Constraint (1) guarantees that the amount of space used in each shelf is at
most $\Delta$ and constraint (2) guarantees that the amount of space used in each
bin is at most $B$. Constraint (3) guarantees that variables $x_{j}^{ic}$ are not greater
than the total size of small items.

Given a packing $\mathcal{P}$, and a set $S$ of small items, the algorithm SMALL first
solves the linear program LPS, and then packs small items in the following
way: For each variable $x_{j}^{ic}$ it packs, while possible, the small items of class $c$
into the shelf $N_{j}^{ic}$, so that the total size of the packed small items is at most $x_{j}^{ic}$.
The possible remaining small items are grouped by classes and packed using
the algorithm SFF into new bins. The complexity time of algorithm SMALL
is polynomial in $n$, since the linear program LPS can be solved in polynomial
time and the algorithm SFF also has polynomial time.

The following lemma is valid for the algorithm SMALL.

**Lemma 3.6** Let $\mathcal{P}$ be a shelf packing of a list of items $L$, where each bin of
$\mathcal{P}$ has at most $\frac{2}{\varepsilon} + 2$ shelves of a same class, $G$ be the set of items in $L$ with
size at least $\varepsilon^2$ and $S$ be the set $L \setminus G$. Let $G'$ be a list of items with $G' \leq G$
and $\mathcal{P}$ be a packing of the items $G' \cup S$ obtained from $\mathcal{P}$ as follows:

1. Let $\mathcal{P}_1$ be the packing obtained from $\mathcal{P}$ removing the items of $S$.
2. Let $\mathcal{P}_2$ be the packing of $G'$ using the algorithm $\text{Alg}_R$ over the pair $(\mathcal{P}_1, G')$.
3. Let $\hat{\mathcal{P}}$ be the packing obtained applying the algorithm SMALL over the
pair $(\mathcal{P}_2, S)$.

Then, we have $|\hat{\mathcal{P}}| \leq (1 + 8C_{\varepsilon})|\mathcal{P}| + C + 1$.

**Proof.** Notice that $|\mathcal{P}_2| = |\mathcal{P}|$ and for each shelf $N_j$ in a bin of $\mathcal{P}$, its
corresponding shelf $N'_j$ in $\mathcal{P}_2$ is such that $s(N'_j) \leq s(N_j)$. If $|\hat{\mathcal{P}}| = |\mathcal{P}|$ then the
lemma follows. So assume that the algorithm SMALL uses additional bins to
pack the items of $S$.

Given a bin $E$, denote by $ns(E)$ the number of shelves in $E$, $ns_{c}(E)$ the number
of shelves of class $c$ in $E$, $ss(E)$ the total size of small items in $E$ and $ss_{c}(E)$
the total size of small items of class $c$ in $E$.

Consider the linear program LPS. An optimum solution for LPS leads to an
optimal fractional packing $\mathcal{P}^*$ of the small items such that $|\mathcal{P}^*| = |\mathcal{P}|$. Consider
a bin $P_{i}^*$ of $\mathcal{P}^*$ and $\hat{P}_{i}$ the corresponding bin in $\hat{\mathcal{P}}$. We first prove that the
following inequality is valid,

$$ss(P_{i}^*) - ss(\hat{P}_{i}) \leq 4C_{\varepsilon}. \quad (13)$$

To prove (13), notice that $ns_{c}(P_{i}^*) = ns_{c}(\hat{P}_{i})$ for each class $c$. Given a shelf $N_{j}^{ic}$
and the corresponding variable $x_{j}^{ic}$, the algorithm SMALL packs a set of
items $T^i_j$ in $N^i_j$ such that $x^i_j - s(T^i_j) \leq \varepsilon^2$ since a small item has size at most $\varepsilon^2$. Since each bin in $P^*$ has at most $\frac{2}{\varepsilon} + 2$ shelves of a same class, we have for each class $c \in [C]$

$$ss_c(\hat{P}) \geq \sum_{j=1}^{n_c}(x^i_j - \varepsilon^2) = ss(P^*) - ns_c(P^*)\varepsilon^2$$

$$\geq ss_c(P^*) - \left(\frac{2}{\varepsilon} + 2\right)\varepsilon^2 \geq ss_c(P^*) - 4\varepsilon.$$

Since the above inequalities are valid for each class we can conclude the proof of (13). From (13), we know that the total size of small items packed in additional bins by SMALL with the algorithm SFF, is at most $4C\varepsilon |P^*| = 4C\varepsilon |P|$. Denote by $\hat{Q}$ the set of additional bins. Each shelf generated by the algorithm SFF is filled by at least $\Delta - \varepsilon^2$, except perhaps in $C$ shelves. Therefore, the number of shelves in $\hat{Q}$ is at most $\left\lceil\frac{4C\varepsilon |P|}{1 - \varepsilon^2}\right\rceil + C \leq 8C\varepsilon |P| + C + 1$. Since each additional bin has at least one shelf, the number of bins in $\hat{Q}$ is at most $8C\varepsilon |P| + C + 1$. Therefore, the number of bins in $\hat{P}$ is at most $(1 + 8C\varepsilon)|P| + C + 1$. □

3.2.3 Analysis of the Algorithm ASBP$_\varepsilon''$

In this section we conclude the analysis of the algorithm ASBP$_\varepsilon''$. First, let $T_{LR}$ and $T_S$ be the time complexities of algorithms $A_{LR}$ and SMALL, respectively. Notice that the time complexity of algorithm ASBP$_\varepsilon''$ is dominated by steps 2–4, that have time complexity $O(T_{LR} + T_{LR}T_S)$. Since the time complexity of algorithms $A_{LR}$ and SMALL is polynomial for fixed $\varepsilon$, the time complexity of algorithm ASBP$_\varepsilon''$ is also polynomial. The following lemma concludes the analysis of the algorithm ASBP$_\varepsilon''$.

**Lemma 3.7** The algorithm ASBP$_\varepsilon''$, is an APTAS for the CCSBP problem when the given instance $I$ is such that $\Delta = 1, \varepsilon \leq (d + \Delta)/B$ and the number of different classes is bounded by some constant $C$.

**PROOF.** Given an instance $I = (L, s, c, d, \Delta, B)$, with $\Delta = 1$, let $G$ be the set of items in $L$ with size at least $\varepsilon^2$ and $S$ the set $L \setminus G$.

The items in $G$ are packed by the algorithm $A_{LR}$. It first partitions $G$ into lists $G_c$ for each class $c$ and then it partitions each list $G_c$ into groups $G^1_c \supseteq G^2_c \supseteq \ldots \supseteq G^k_c$. From Lemma 3.5 the following inequality is valid for the list $G^1 = \bigcup_{c=1}^G G^1_c$.

$$|P_1| \leq 4\varepsilon \OPT(I) + 1. \quad (14)$$
The packing of the items in $G_1^2 || \ldots || G_1^k || \ldots || G_C^2 || \ldots || G_C^{k_C}$ is obtained from the set of all possible packings of $\overline{G_1^2} || \ldots || \overline{G_1^{k_1-1}} || \ldots || \overline{G_C^2} || \ldots || \overline{G_C^{k_C-1}}$. Notice that

$$\overline{G_1^2} || \ldots || \overline{G_1^{k_1-1}} || \ldots || \overline{G_C^2} || \ldots || \overline{G_C^{k_C-1}} \geq G_1^2 || \ldots || G_C^{k_C}.$$

Let $\mathcal{O}$ be an optimum shelf packing of $I$, $\mathcal{O}_1$ the packing obtained from $\mathcal{O}$ without the items of $S$ but with the possible empty shelves and $\mathcal{O}_2$ the packing of $\mathcal{O}_1$ rounding down each item size to the corresponding item in

$$\overline{G_1^2} || \ldots || \overline{G_1^{k_1-1}} || \ldots || \overline{G_C^2} || \ldots || \overline{G_C^{k_C-1}}.$$

Clearly, $\mathcal{O}_2 \in \mathbb{P}$, where $\mathbb{P}$ is the set of packings generated by the algorithm $A_{\text{ALL}}$. Let $\hat{\mathcal{O}}$ be a packing obtained from the algorithm $A_{\text{R}}$ over the pair

$$(\mathcal{O}_2, G_1^2 || \ldots || G_1^{k_1} || \ldots || G_C^2 || \ldots || G_C^{k_C}).$$

If $\mathcal{Q}$ is a packing obtained applying the algorithm SMALL over the pair $(\hat{\mathcal{O}}, S)$, we have from Lemma 3.6 the following result.

$$\mathcal{Q} \leq (1 + 8C \varepsilon) |\mathcal{O}| + C + 1 = (1 + 8C \varepsilon) \text{OPT}(I) + C + 1 \quad (15)$$

Since the algorithm $\text{ASBP}_\varepsilon''$ obtains a packing $\mathcal{P}$ that uses at most the number of bins in $\mathcal{P}_1 \cup \mathcal{Q}$, the theorem follows from inequalities (14) and (15). \hfill \Box

From lemmas 3.2 and 3.7, the following statement holds.

**Theorem 3.8** The algorithm $\text{ASBP}_\varepsilon$ is an APTAS for the CCSBP problem.

### 4 Concluding Remarks

In this paper we consider the CCSBP problem, a class constrained bin packing problem with non-null shelf divisions. Although this problem has many practical applications, to our knowledge, this is the first paper to present approximation results for it. We first presented hybrid versions of the First Fit (Decreasing) and Best Fit (Decreasing) algorithms for the bin packing problem to the CCSBP problem. When the number of different classes of items is bounded by a constant $C$, we prove that the versions of the First Fit and Best Fit have asymptotic performance bound 3.4 and the versions of the First Fit Decreasing
and Best Fit Decreasing have asymptotic performance bound 2.445. We also presented an APTAS for this same case whose running time is

\[ O(n^{O(2/\epsilon)^{O(1/\epsilon^2); C/\epsilon^3}}). \]

This algorithm is more of theoretical (rather than practical) interest since it has a high running time (yet polynomial). When the number of classes is not bounded by a constant we show that the algorithm SFFD has absolute performance bound 3.

5 Acknowledgements

We would like to thank the anonymous referees for the helpful suggestions which improved the presentation of this paper.

References


